UNIVERSIDADE FEDERAL DE VIÇOSA
CAMPUS RIO PARANAÍBA
SISTEMAS DE INFORMAÇÃO

LUCAS NARDELLI DE FREITAS BOTELHO SAAR

UMA APLICAÇÃO MÓVEL PARA CONSUMO DE SERVIÇOS COGNITIVOS EM UMA ARQUITETURA DE INTELIGÊNCIA ARTIFICIAL COMO SERVIÇO

RIO PARANAÍBA
2023

LUCAS NARDELLI DE FREITAS BOTELHO SAAR

UMA APLICAÇÃO MÓVEL PARA CONSUMO DE SERVIÇOS COGNITIVOS EM UMA ARQUITETURA DE INTELIGÊNCIA ARTIFICIAL COMO SERVIÇO

Monografia apresentada à Universidade Federal de Viçosa, como parte das exigências para a a aprovação na disciplina Trabalho de Conclusão de Curso II.

Orientador: Dr. Rodrigo Moreira.

RIO PARANAÍBA

2023

LUCAS NARDELLI DE FREITAS BOTELHO SAAR

UMA APLICAÇÃO MÓVEL PARA CONSUMO DE SERVIÇOS COGNITIVOS EM UMA ARQUITETURA DE INTELIGÊNCIA ARTIFICIAL COMO SERVIÇO

Monografia apresentada à Universidade Federal de Viçosa, como parte das exigências para a a aprovação na disciplina Trabalho de Conclusão de Curso II.

APROVADA EM: 11 de dezembro de 2023

Dr. Rodrigo Moreira.
Orientador
UFV-CRP

Membro da Banca A
UFV-CRP

Membro da Banca B
UFV-CRP

RIO PARANAÍBA
2023

Dedico este trabalho aos meus pais, cujo esforço incansável tornou possível a conclusão do meu curso. Cada página é um reflexo do amor e dedicação que me proporcionaram. Este trabalho é para vocês, que são a razão pela qual estou aqui hoje.

Agradecimentos

Gostaria de expressar minha profunda gratidão a todos que contribuíram para a realização deste trabalho, que representa o culminar de anos de estudo e dedicação.

Primeiramente, quero agradecer de maneira especial ao meu orientador, Rodrigo Moreira, pela orientação excepcional, paciência e apoio constante ao longo deste processo. Sua expertise e insights foram fundamentais para a conclusão deste trabalho, e sou imensamente grato pela oportunidade de aprender com alguém tão dedicado e inspirador.

Agradeço também a todos os professores que contribuíram para o meu crescimento acadêmico. Cada aula, cada conselho e cada desafio moldaram meu entendimento e ampliaram minha perspectiva sobre o tema deste trabalho.

Não posso deixar de mencionar minha imensa gratidão aos meus familiares e amigos. Seu apoio inabalável, encorajamento e compreensão foram essenciais para que eu superasse desafios e mantivesse o foco na realização deste projeto. Agradeço por cada palavra de incentivo, gesto de carinho e compreensão nos momentos mais intensos desta jornada.

Este trabalho não seria possível sem a colaboração e apoio de cada pessoa mencionada. A todos vocês, o meu mais sincero obrigado.

Resumo

Com o crescimento contínuo do número de dispositivos móveis, observa-se um considerável aumento na produção de dados, especialmente no que se refere à quantidade de imagens. Esse cenário despertou um interesse significativo no processamento e classificação eficientes dessas imagens. Diante da diversidade dessas imagens e da ausência de um aplicativo genérico abrangente para lidar com diversas áreas de classificação de imagem, este trabalho propõe a criação de uma solução que atenda a essa necessidade. A proposta consiste em um aplicativo capaz de interagir com uma Arquitetura de Inteligência Artificial como Serviço (IAaaS) que inclui, entre suas funcionalidades, uma loja de modelos de Inteligência Artificial (IA) (denominado *Models Store*). Essa loja abriga uma variedade de modelos de aprendizado de máquina previamente treinados, disponíveis para consumo sob o modelo "pay as you go". Para viabilizar este projeto, desenvolvemos um aplicativo que se comunica com a Interface de Programação de Aplicação (API) da *Models Store*, permitindo a solicitação e utilização de modelos específicos. Após o desenvolvimento, conduzimos uma série de testes de desempenho, analisando parâmetros como tempo de predição, consumo de memória e utilização da unidade central de processamento (CPU). Este trabalho resultou na criação do aplicativo mencionado, estabelecendo um meio eficaz de comunicação com a loja de modelos, e proporcionando uma comparação entre a execução dos modelos no dispositivo móvel e em um servidor de alta performance.

Palavras-chaves: Classificação de imagem, Models Store, IA como serviço, Aplicativo Móvel.

Abstract

With the continuous growth in the number of mobile devices, there has been a considerable increase in data production, particularly concerning the quantity of images. This scenario has sparked significant interest in efficient image processing and classification. Given the diversity of these images and the absence of a comprehensive generic application for handling various image classification areas, this work proposes the creation of a solution to address this need. The proposal entails an application capable of interacting with an IAaaS Architecture, which includes a Models Store as one of its functionalities. This store hosts a variety of pre-trained machine learning models available for consumption under the "pay as you go" model. To realize this project, we developed an application that communicates with the Models Store API, allowing the request and utilization of specific models. Following development, a series of performance tests were conducted, analyzing parameters such as prediction time, memory consumption, and CPU utilization. This work resulted in the creation of the mentioned application, establishing an effective means of communication with the models store, and providing a comparison between model execution on a mobile device and a high-performance server.

Key-words: Image classification, Models Store, AI as a Service, Mobile Application.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Recentemente, com a popularização dos aplicativos móveis, os smartphones se tornaram onipresentes e essenciais em nossas vidas, modificando a forma como nos comunicamos, trabalhamos e acessamos informações. Estima-se que o número de pessoas possuindo seu próprio celular atinja a casa de 5,8 bilhões no ano de 2025 cujo número corresponde a cerca de 71% da população mundial (VALENTE, 2019). Este crescimento acelerado tem sido impulsionado pela praticidade, acessibilidade e funcionalidade que oferecem permitindo que tenhamos acesso a diversos serviços e proporcionando uma experiência personalizada, adaptando-se as preferências de cada usuário (NUNES, 2021).

Com este aumento no número de dispositivos, consequentemente houve um grande aumento no número de imagens, gerando uma oportunidade de analisar estas novas imagens. A área de visão computacional é a responsável pela interpretação e classificação destas imagens através de algoritmos computacionais. Porém existem diversos campos diferentes de classificação de imagem incluindo segurança, reconhecimento facial, diagnósticos médicos e entre outros (SCABINI et al., 2017).

Tendo em vista que os usuários dos aplicativos móveis esperam que estes possam adaptar-se às suas necessidades, é possível identificar uma carência de um *app* genérico que utiliza o paradigma Inteligência Artificial como Serviço (IAaaS) com o objetivo de viabilizar o usuário conseguir selecionar qual algoritmo de Inteligência Artificial (IA) gostaria de utilizar, possibilitando assim classificar diversos tipos de imagens diferentes em um mesmo aplicativo, evitando desta forma que precise instalar um aplicativo para cada problema que deseje tratar (RODRIGUES MOREIRA et al., 2023).

Para o correto funcionamento do aplicativo é necessário que existam três entidades a primeira sendo uma *Models Store*, na qual possui diversos modelos de IA disponíveis para classificação de imagem, a segunda seria uma Interface de Programação de Aplicação (API) para prover a comunicação entre as duas outras entidades e a terceira engloba o aplicativo. A sequência de funcionamento do *app* é realizada pelo usuário se autenticando no sistema em seguida selecionando qual modelo de IA utilizará, deste modo será requisitado à loja de modelos para que em seguida ocorra a classificação da imagem e exiba o resultado na tela do aplicativo.

O presente estudo envolveu o desenvolvimento de um aplicativo compatível com os sistemas operacionais Android e iOS, no qual realizou-se parte do conceito IAaaS que é a busca por modelos já treinados para predição local de diferentes tipos de doenças. Posteriormente, por meio da execução de testes de desempenho, observou-se que a abordagem de predição *in situ* no dispositivo revelou-se mais vantajosa em termos de tempo de

previsão, consumo de memória e utilização da unidade central de processamento (CPU).

O presente trabalho está organizado da seguinte maneira Capítulo 2 apresenta os objetivos a serem atingidos no desenvolvimento do projeto. O Capítulo 3 discute sobre alguns conceitos importantes para a compreensão do funcionamento do aplicativo, sendo computação em nuvem, aprendizado de máquina, redes neurais e aborda sobre as ferramentas computacionais utilizadas no desenvolvimento do aplicativo. Em seguida, no Capítulo 4, é realizada uma revisão bibliográfica discorrendo sobre certos trabalhos relacionados ao presente. Os métodos de desenvolvimento são abordados no Capítulo 5, no qual mostra as telas e os principais diagramas que explicam o funcionamento do aplicativo. Nos Capítulos 6 e 7 são abordados os resultados para a implementação e a conclusão do trabalho.

2 Objetivos

O objetivo geral do trabalho é propor, construir e avaliar uma abordagem de recuperação *online* de modelos de aprendizado de máquina para classificação de imagens. Suprindo uma carência existente de um aplicativo genérico para realizar a classificação de diversos tipos diferentes de imagem. O funcionamento do projeto é implementado no desenvolvimento desse aplicativo genérico, no qual através de uma API consegue realizar a comunicação com uma *Models Store* cuja possui diversos modelos de IA previamente treinados e pronto para serem usados na classificação de imagem. Desta maneira se tornará possível que um usuário se sinta confortável podendo realizar a instalação somente do aplicativo desenvolvido no presente trabalho para obter um resultado da classificação de imagem de sua escolha.

2.1 Objetivos Específicos

Recuperar modelos de aprendizado de máquina da nuvem para classificação de imagens requer satisfazer os seguintes objetivos.

- **Um aplicativo genérico para classificação de imagens baseada em IA.**
- **Um método para recuperação de modelos de aprendizado de máquina na *Models Store*.**
- **Um caso de uso de recuperação de um modelo de IA para classificação de frutos do café.**
- **Aplicar um teste de desempenho no aplicativo.**

3 Referencial Teórico

Para desenvolvimento da presente proposta, alguns conceitos e técnicas são necessárias. Para isso, esta seção descreve computação em nuvem na Seção 3.1, aprendizado de máquina na Seção 3.2, redes neurais na Seção 3.3 e é discutido sobre as ferramentas computacionais utilizadas no desenvolvimento do projeto na Seção 3.4.

3.1 Computação em Nuvem

Antes da computação em nuvem as empresas eram obrigadas a adquirir recursos computacionais de acordo com a demanda exigida por suas operações, considerando que tal demanda nunca é linear acontece de, em algum momento, essa infraestrutura ficar ociosa. Este desperdício de recursos é um grande problema (PEDROSA; NOGUEIRA, 2011).

A computação em nuvem tem como prerrogativa justamente resolver este problema, na qual consiste em provisionar infraestrutura de Tecnologia da informação (TI) onipresente através da internet ou por rede, sob demanda. Seu funcionamento se dá pelas empresas assinando um contrato com o provedor de computação em nuvem, no qual disponibilizará uma abstração dos componentes computacionais em que será tarifado de acordo com o uso, mitigando gastos quando seus servidores ficarem ociosos (PEDROSA; NOGUEIRA, 2011).

Logo, computação em nuvem é a disponibilização de recursos de tecnologia através da internet, onde as empresas contratantes os acessam de maneira remota e não precisam ter conhecimento técnico necessário para construir toda essa infraestrutura fisicamente. Toda a parte de TI é provisionada pela empresa fornecedora da computação em nuvem, de forma que o cliente só precisa pagar pelo tempo e quantidade de recursos utilizados. Este modelo de computação faz parecer que existe infraestrutura infinita, pois é alocado recursos de acordo com a quantidade de requisição (PEREIRA; SACILOTTI; JÚNIOR, 2019).

Para implementar a computação em nuvem é possível realizar por meio de quatro modelos: nuvem privada, pública, comunitária e híbrida. A nuvem privada é utilizada por apenas uma empresa e é feita sob demanda para atender suas especificações sendo totalmente administrada pela mesma. A nuvem pública é compartilhada por diversos usuários finais, onde é fornecido serviços simultaneamente e cada usuário não pode decidir quem tem acesso a esse serviço. Nuvem comunitária tem as mesmas características da nuvem pública, o diferencial é que os usuários compartilham do mesmo propósito e ideais. A nuvem híbrida é uma junção de pelo menos dois modelos citados anteriormente (SUNYAEV,

2020).

Computação em nuvem pode ser descrita em três classes infraestrutura como serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS), que será discorrido sobre cada uma delas nas seções seguintes. Sendo classificadas de acordo com o nível de abstração e serviço provido. A Figura 1 é uma exemplificação de como essas três classes são divididas na computação em nuvem.

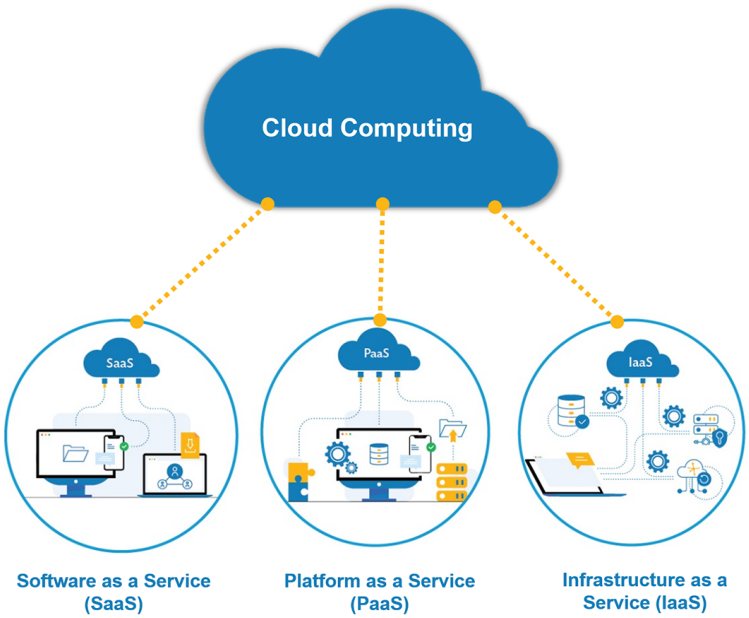

Figura 1 – Ilustração representativa das camadas da computação em nuvem

.

Fonte: Abell (2021)

3.1.1 IaaS

Infraestrutura como serviço é a disponibilização de recursos de processamento, armazenamento e rede. É realizado oferecendo uma máquina virtual para o cliente, onde ele pode customizar essa máquina escolhendo o sistema operacional e os aplicativos instalados. porém o cliente da computação em nuvem não tem o poder de controlar a infraestrutura da nuvem. A Amazon Web Services (AWS), Microsfot Azure e Digital Ocean são exemplos de IaaS (SUNYAEV, 2020).

3.1.2 PaaS

Plataforma como serviço os usuários podem instalar na nuvem aplicativos criados ou adquiridos usando linguagens de programação que sejam compatíveis com a nuvem. Neste modelo o cliente não tem o mesmo acesso da IaaS para configurar sistema ope-

racional, somente têm acesso a configuração de aplicativos e do ambiente. Temos como exemplos de PaaS a Google App Engine, Windows Azure e Heroku (SUNYAEV, 2020).

3.1.3 SaaS

No *software* como serviço a empresa responsável por este modelo disponibiliza aplicativos completos para seus clientes onde a maior parte do processamento ocorre na nuvem. Estes clientes apenas tem a possibilidade de alterar algumas poucas configurações do aplicativo em questão. Podemos citar como exemplo de SaaS a Xbox Cloud Gaming, DropBox e Google WorkSpace (SUNYAEV, 2020).

3.2 Aprendizado de Máquina

O surgimento do conceito de aprendizado de máquina foi realizado pelo psicólogo Frank Rosenblatt em 1957, na universidade de Cornell. No qual tendo como base o funcionamento do sistema nervoso humano, que juntamente com um grupo, fabricou uma máquina capaz de reconhecer letras do alfabeto. Esta recebeu no nome de "perceptron" na qual recebia sinais analógicos e os convertia em sinais discretos (FRADKOV, 2020).

No ano de 1959, um professor de Stanford e pesquisador da International Business Machines Corporation (IBM), Arthur L. Samuel declarou que o *Aprendizado de Máquina (ML)* é o "campo de estudo que dá aos computadores a capacidade de aprender sem serem explicitamente programados". O *ML* é um subconjunto de IA com foco em fazer computadores produzirem resultados sem explicitamente programá-los (MARKIEWICZ; ZHENG, 2020).

A IA pode ser compreendida como uma técnica na qual os computadores possam reproduzir o comportamento humano para a resolução de problemas complexos. Por sua vez o *ML* retrata a habilidade do computador de ser treinado com base de dados, para que de forma automática, ele seja capaz de construir modelos analíticos que concebem previsões, regras, respostas ou recomendações (JANIESCH; ZSCHECH; HEINRICH, 2021).

É concebível dizer que um *software* aprendeu se, com sua experiência, são obtidos resultados melhores e mais precisos à medida que são fornecidos novos dados de treinamento para a IA em questão. Logo este programa de computador toma decisões e faz previsões totalmente baseado nos dados que a ele foram fornecidos. Essas decisões são tomadas a fim de resolver algum dos três tipos de problemas: classificação, regressão e agrupamento (RAY, 2019).

Portanto, o *ML* propõe automatizar a criação de modelos analíticos para a execução de trabalhos cognitivos e o desempenho de um *software* é aprimorado diretamente a partir de sua experiência. Tendo sua automatização obtida com a aplicação de algoritmos iterativos, permitindo que sistemas encontrem percepções e padrões complexos

(JANIESCH; ZSCHECH; HEINRICH, 2021). Estes algoritmos estão compreendidos dentro de três classes de aprendizado: Aprendizado supervisionado 3.2.1, aprendizado não supervisionado 3.2.2 e aprendizado por reforço 3.2.3.

Na Figura 2 é exemplificado o processo do aprendizado de máquina que foi discutido anteriormente. Onde primeiramente têm-se a fase de aquisição dos dados para após fazer um processamento desses dados, ou seja, limpar e melhorar a qualidade desses dados. Posteriormente, na fase de processo, é aplicado os algoritmos de *ML* que no final de seu processo produzirão os resultados.

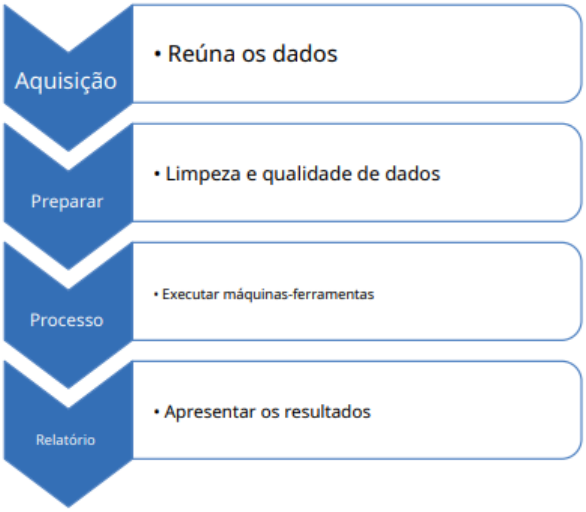

Figura 2 – Ilustração para representar o processo completo do aprendizado de máquina
.

Fonte: Bell (2007)

3.2.1 Aprendizado supervisionado

O aprendizado supervisionado consiste em treinar uma IA, com um conjunto de dados previamente estruturados e rotulados. Basicamente quanto maior e melhor for o seu conjunto de dados de entrada melhor será o resultado obtido pelo seu algoritmo. Ainda sobre o tipo de aprendizado supervisionado podemos listar duas problemáticas na qual ele resolve, a de regressão e a de classificação (JANIESCH; ZSCHECH; HEINRICH, 2021).

3.2.2 Aprendizado não supervisionado

Diferentemente do citado anteriormente, no aprendizado não supervisionado não é fornecido dados estruturados e rotulados. Para seu funcionamento apenas é necessário um conjunto de dados de treinamento que compreendem variáveis, tendo como finalidade estruturar os dados de acordo com o interesse. Este tipo de aprendizado pode ser utilizado

para tarefas de agrupamento e redução de dimensionalidade (JANIESCH; ZSCHECH; HEINRICH, 2021).

3.2.3 Aprendizado por reforço

Na aprendizagem por reforço é informado o estado do sistema, suas restrições e as metas a serem atingidas, ou seja, não é fornecido pares de entrada e saída. É tarefa do algoritmo, utilizando a regra de tentativa e erro, buscar atingir a meta que foi fornecida a fim de maximizar sua recompensa (JANIESCH; ZSCHECH; HEINRICH, 2021).

3.3 Redes Neurais

Em 1943, McCullock e Pitts realizaram o primeiro trabalho envolvendo o tema de Redes Neurais, no qual conduziram um estudo a fim de desenvolver um modelo matemático ao observar o funcionamento do neurônio biológico. Outro pesquisador de extrema importância foi Frank Rosenblatt, com a criação do perceptron, cujo foi abordado na Seção 3.2. Porém este trabalho foi fortemente criticado no livro "Perceptrons", dos autores Minsky e Papert, que abordam a deficiência do perceptron em simular uma simples função de ou exclusivo (XOR). Depois de muito tempo, em 1972, Teuvo Kohonen propôs um novo modelo de rede neural utilizando um conjunto de neurônios que interagem entre si, introduzindo o conceito de aprendizado não supervisionado nas Redes Neurais Artificiais (RNAs). Uma nova evolução foi realizada por John Hopfield em 1982, onde foi desenvolvido o modelo de redes com retroalimentação (CARDON; MÜLLER; NAVAUX, 1994).

As RNAs são um sistema desenvolvido para modelar computacionalmente a forma como um cérebro funciona por meio de simulação em uma máquina. Ou seja, são algoritmos desenvolvidos de forma a apresentar um modelo matemático inspirado no cérebro humano, tornando assim a Rede Neural Artificial (RNA) capaz de aprender e tomar decisões de forma autônoma, dependendo apenas de seu aprendizado anterior. É perceptível que a rede neural tem características em comum com o cérebro como adquirir conhecimento de acordo com suas experiências de aprendizagem e possuir conexões entre os nós de processamento (neurônios) (FLECK et al., 2016).

Abordando ainda as redes neurais, podemos separá-las em quatro componentes, sendo dois físicos (elemento de processamento e conexões) e dois não físicos (Padrão e funções). O elemento de processamento é o neurônio, onde ocorre todo o processamento da entrada recebida para se produzir uma única saída. As conexões são os componentes que conectam os neurônios, transportando sempre a informação da saída de um para a entrada de outro elemento de processamento ou para fora do sistema. O padrão nada mais é do que o dado de entrada para a rede neural. E por último as funções são os

modelos matemáticos presentes na RNA para reconhecer padrões e realizar o treinamento (CARDON; MÜLLER; NAVAUX, 1994).

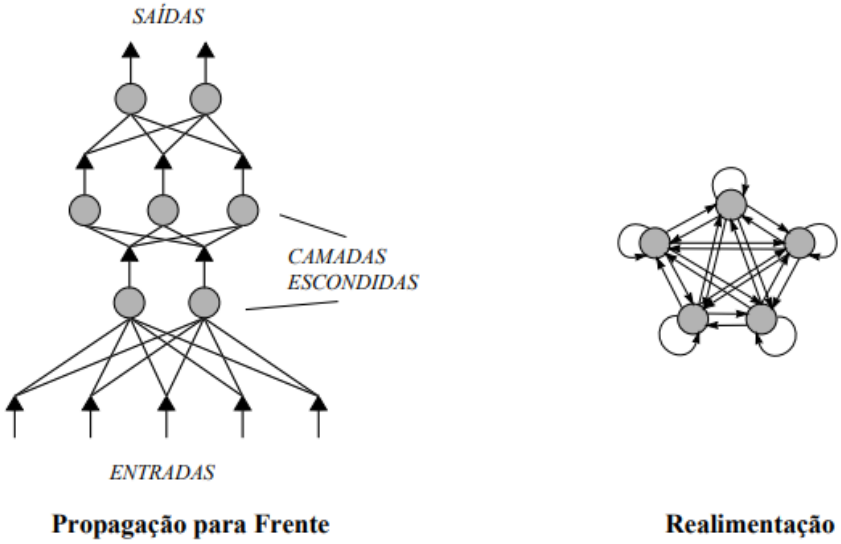

Figura 3 – Ilustração para representar o funcionamento de uma RNA de Propagação para Frente e de Realimentação

.

Fonte: Rauber (2005)

Podemos distinguir tipos de RNAs de acordo com o fluxo e propagação das informações realizadas. Como ilustrado na Figura 3 são destacados dois tipos a RNA com propagação para frente que consiste no fluxo de informações unidirecional tendo os neurônios que recebem estes dados sendo agrupados em camadas, no qual as camadas que não estão diretamente ligadas com as saídas ou entradas recebem o nome de camadas escondidas. As RNAs com realimentação não utilizam um fluxo de informação unidirecional, mas exercem um comportamento dinâmico possuindo conexões sem restrições entre os neurônios (RAUBER, 2005).

De acordo com Haykin (2001) podemos destacar algumas vantagens no uso das redes neurais como a adaptabilidade, na qual elas são capazes de adequar seus pesos de acordo com a modificação do ambiente, tendo também uma boa resposta a evidências retornando informações sobre a confiança ou crença na decisão tomada e uma RNA implementada em hardware é tolerante a falhas. Porém é possível perceber desvantagens em relação aos dados de entrada que precisam ser tratados previamente e exigindo um grande volume destes dados, tornando o treinamento demorado.

3.4 Ferramentas Computacionais

A API fornece uma forma de estabelecer comunicação entre dois *softwares* diferentes, permitindo que estas aplicações troquem informações entre si, podendo estas serem de empresas diferentes. API pode ser definida como uma coleção de código, permitindo dessa maneira a reutilização de funções previamente implementadas e melhorando a produtividade dos desenvolvedores. É uma forma do programador consumir informações e serviços de outras empresas dentro de seu próprio aplicativo (OFOEDA; BOATENG; EFFAH, 2019).

React Native é uma linguagem de programação baseada em React, que é a biblioteca JavaScript do Facebook para construir interfaces de usuário. O React Native é implementado para a construção de aplicativos moveis, para as plataformas Sistema Operacional do IPhone (IOS) e Android. Esta linguagem é escrita usando JSX que é uma mistura de Extensible Markup Language (XML) com JavaScript, em seguida o React Native invoca a API nativa Objective-C para IOS e JAVA para android, é possível visualizar um exemplo deste processo na Figura 4 (EISENMAN, 2015).

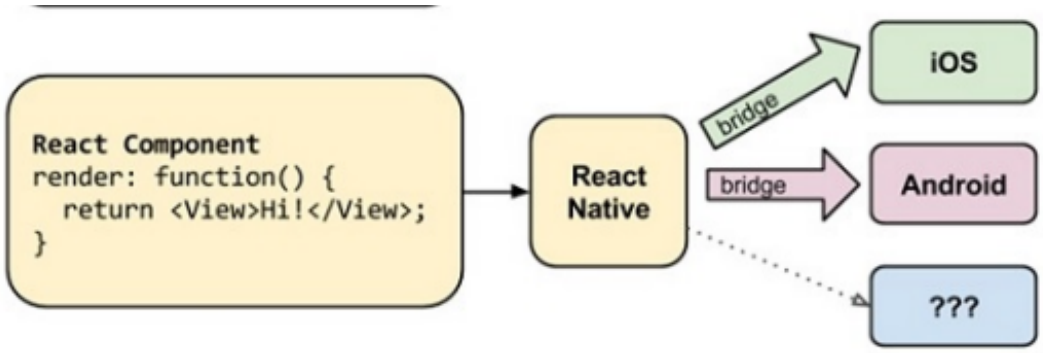

Figura 4 – Exemplificação do processo de invocação das APIs nativas

.

Fonte: Danielsson (2016)

Neste capítulo foram discutidos conceitos que fundamentam a presente proposta. O conceito de IAaaS objetiva facilitar a entrega de serviços cognitivos para dispositivos móveis, permitindo a classificação de diferentes imagens em dispositivos de borda (*edge*) (RODRIGUES MOREIRA et al., 2023). Essa entrega se dá pela materialização do conceito *Models Store* que é a funcionalidade onde os usuários, com diferentes demandas, podem adquirir para seu dispositivo móvel um modelo de aprendizado de máquina customizado e já treinado. Assim, o usuário pode realizar a classificação conforme sua demanda.

4 Trabalhos Relacionados

Neste capítulo, serão descritos os trabalhos relacionados com o proposto. Ao final, é apresentada uma tabela que visa comparar o trabalho proposto.

O trabalho de Zeng et al. (2022) apresenta uma abordagem inovadora na concepção de uma rede neural convolucional de menor densidade, visando facilitar e otimizar o processo de classificação de imagens. Este modelo, desenvolvido com o propósito de integração em dispositivos móveis, demonstra um enfoque direcionado à eficiência no reconhecimento de doenças nas folhas de milho. A implementação foi realizada mediante o treinamento da rede utilizando uma base de dados específica relacionada às folhas de milho.

O estudo conduzido por Moreira et al. (2022) introduz uma proposta de arquitetura de baixo custo destinada à classificação da saúde das folhas de milho em ambientes agrícolas. Essa arquitetura abrange a integração de sensores responsáveis pela aquisição de imagens, um servidor dedicado ao processamento das imagens e, por fim, um aplicativo móvel que viabiliza a visualização das predições geradas. Este conjunto inovador de tecnologias representa um avanço significativo no processo de identificação de folhas doentes no contexto do agronegócio, contribuindo para a evolução das práticas agrícolas rumo à implementação de fazendas inteligentes.

Por outro lado, Esgario et al. (2022) propõe a criação de um aplicativo destinado à avaliação da saúde das folhas de café. O autor conduziu um treinamento de uma rede neural utilizando um conjunto de dados composto por informações sobre o café. O modelo gerado tem como finalidade identificar diversas doenças, incluindo *"leaf miner"*, *"rust"*, *"brown leaf spot"* e *"cercospora leaf spot"*. Essa abordagem busca fornecer uma ferramenta prática para a identificação e classificação eficiente das condições de saúde das folhas de café, contribuindo para a gestão e prevenção de doenças que afetam essa cultura.

É relevante salientar que este trabalho se distingue dos anteriores ao não se restringir exclusivamente à classificação de doenças em folhas de milho ou café. Através da integração de uma loja de modelos, o aplicativo desenvolvido apresenta a capacidade de incorporar uma variedade de algoritmos de aprendizado de máquina, tornando-se assim mais versátil e apto a abordar diferentes desafios em cenários agrícolas. Essa flexibilidade ampliada destaca o potencial do sistema para além da sua aplicação inicial, conferindo-lhe adaptabilidade a diversas problemáticas.

O trabalho proposto por Kim et al. (2021) discute a possibilidade de pessoas conseguirem adquirir um diagnóstico de sua saúde bucal em suas próprias casas, a partir de um dispositivo criado capaz de obter imagens dos dentes superiores e inferiores do

paciente e em seguida com a aprendizagem profunda classificar se este usuário necessita de tratamento especializado. Neste artigo foram utilizados os modelos RetinaNet, You Only Look Once version 3 (YOLOv3) e Faster R-CNN, que após treinamentos obtiveram 97% de precisão na identificação de Região de Interesse (ROI) de dentes e 96% e 89% de precisão na classificação de doenças dentárias e Tratamento Odontológico Profissional (NPDT).

Diferentemente, o objetivo do presente trabalho é aumentar o número de pessoas que tenham acesso a um diagnóstico de sua saúde bucal, para tornar isso possível é proposto que as imagens sejam obtidas a partir do próprio celular do paciente, sem a necessidade de adquirir um equipamento especializado para conseguir registrar estas imagens, tornando assim um pouco mais acessível este diagnóstico em casa e atingindo mais pessoas.

A utilização de uma IA classificando a presença de cavidades dentárias analisando fotos nas quais os próprios usuários são capazes de obter com seus telefones é de fato interessante, porém Bhattacharjee (2022) vai um pouco além e disponibiliza uma explicação de como a IA chegou no veredito do diagnóstico, fazendo com que o paciente se sinta mais confortável em confiar no resultado produzido pelo processamento de imagem. Para a classificação foi utilizado a arquitetura ResNet-27 aplicando o aprendizado curricular em dois estágios e obtiveram uma precisão da identificação de cárie de 82,8% e sensibilidade de 1,0.

É proposto por Thanh et al. (2022) a utilização do celular para capturar fotos da superfície lisa do dente e gerar um possível diagnóstico de cárie para o usuário. Para atingir este objetivo ele usa rede neural artificial de treinamento massivo e rede neural convolucional, os algoritmos de detecção de objetos utilizados são Faster R-CNN, YOLOv3, RetinaNet e Single-Shot Multibox Detector (SSD), tendo como métrica de avaliação a sensibilidade, especificidade e a precisão que no melhor algoritmo atingiu 67%, 79% e 87,4% respectivamente.

O trabalho de Baek, Kim e Shin (2022) foi implementado um algoritmo de YOLOv3 para identificar imagens orais ou não orais com a finalidade de melhorar um aplicativo já existente que classifica a saúde bucal dos pacientes a partir de fotos tiradas pelos mesmos, tal identificação se da por algoritmos que fazem um reconhecimento de objetos, por exemplo este é capaz de identificar dentes humanos na imagem. Porém se o usuário fotografar qualquer coisa que não seja sua boca o aplicativo irá classificar erroneamente com base na imagem, então a proposta seria antes utilizar o algoritmo que reconhece imagens orais para caso de fato seja uma foto dos dentes, prosseguir para a próxima etapa de identificar o estado da saúde bucal.

Por sua vez este artigo pretende disponibilizar a identificação de cárie dental, não levando em conta esta análise prévia se de fato o usuário estaria enviando as devidas fotos. A melhoria que é proposta seria uma abordagem inovadora no aplicativo de habilitar

acesso a uma biblioteca *online* de recuperação de modelos já treinados para classificar diferentes problemas.

Tendo em vista que a utilização de telefones celulares cresce cada vez mais, está tornando possível ter diagnósticos médicos a partir de aplicativos. No Trabalho de Ding et al. (2021) é explorado justamente esta temática onde ele propõe um treinamento de IA utilizando o algoritmo YOLOv3 para identificar cáries nas imagens registradas por celulares dos próprios pacientes. Após treinamento é obtido 69,42% e 93,33% de recordação e precisão respectivamente para cárie primária e para cárie secundária temos 52,38% de recordação e 100% de precisão. Diferente do trabalho desenvolvido, a abordagem de Ding et al. (2021) não é capaz de recuperar os modelos de predição *online*, sendo portanto não compatível com a arquitetura IAaaS.

Para sumarizar a comparação entre os trabalhos relacionados, propõe-se a Tabela 1. A coluna Classificação, objetiva classificar a abordagem quanto o tipo de tarefa que o algoritmo de IA faz. Por sua vez a coluna Recuperação de Modelos aponta se no artigo em questão foi abordado a construção de uma API capaz de recuperar modelos *online* já treinados de classificação de imagens. A coluna Métrica de Qualidade retrata os parâmetros que cada artigo utilizou para medir o sucesso dos resultados obtidos com seus algoritmos. A coluna Algoritmo Utilizado aborda de fato o algoritmo computacional de classificação que foi utilizado para a realização da proposta de cada artigo. A coluna Pré-processamento indica, caso tenha implementado, qual algoritmo para melhorar as imagens obtidas foi implementado.

Tabela 1 – Comparação entre os Trabalho Relacionados

Artigo	Tarefa de IA	Recuperação de Modelos	Métrica de Qualidade	Algoritmo Utilizado	Pré-processamento
Moreira et al. (2022)	●	○	Acurácia, Precisão, Recall e F1-score	Resnet50, MobilenetV2, VGG16 e Efficientnet	○
Zeng et al. (2022)	●	○	Acurácia, Precisão e Recall	LDSNet	○
Esgario et al. (2022)	●	○	Acurácia, Precisão e Recall	AlexNet, GoogLenet e VGG16	○
Kim et al. (2021)	●	○	Precisão, Recall e F1-score	RetinaNet, YOLOv3 e Faster R-CNN	MSRCR
Bhattacharjee (2022)	●	○	Precisão e Sensibilidade	ResNet-18 e ResNet-27	○
Thanh et al. (2022)	●	○	Precisão e Sensibilidade	Faster R-CNN	Gaussiano
Baek, Kim e Shin (2022)	●	○	Exatidão, precisão e Recall	YOLOv3	○
Ding et al. (2021)	●	○	Precisão, Recall e F1-score	YOLOv3	MSRCR
Presente Trabalho	●	●	Precisão, Recall e F1-score	YOLOv3 e Faster R-CNN	○

Esses trabalhos levantados são parte do estado da arte que versão sobre a temática de classificação de doenças utilizando algoritmos de aprendizado de máquina. Das abordagens analisadas, constatou-se que nenhuma delas implementam o conceito de IAaaS.

5 Métodos

Este capítulo apresenta de forma detalhada a estruturação e construção da aplicação, destacando as tecnologias empregadas. Serão explorados a orquestração da arquitetura da aplicação móvel, fornecendo exemplos da prototipação das telas. Além disso, aborda-se o gerenciamento dos modelos de aprendizado de máquina, detalhando o processo envolvido nessa etapa crucial do desenvolvimento. Ao final destrinchamos o cenário utilizado nos experimentos.

5.1 Desenvolvimento e Tecnologias

A construção de um aplicativo móvel para o consumo de IAaaS envolve uma série de passos essenciais, conforme ilustrado na Figura 5. O dispositivo móvel, representado na parte inferior esquerda, é o local onde o aplicativo de classificação de imagem é instalado, permitindo a interação do usuário final na escolha de um modelo de IA adequado a seus objetivos, disponível na loja de modelos. A comunicação entre o aplicativo no dispositivo móvel e a loja de modelos na nuvem ocorre no centro da representação, facilitada por uma API capaz de recuperar modelos *online*. À direita, a loja de modelos armazena na nuvem todos os modelos de IA previamente treinados e prontos para uso.

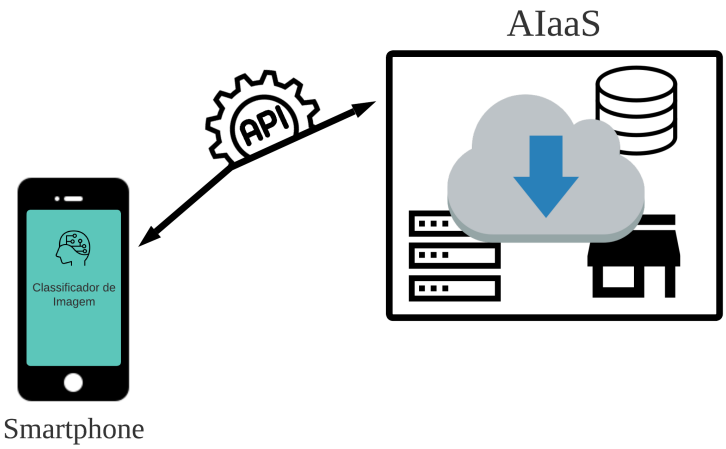

Figura 5 – Imagem mostrando a arquitetura do projeto

.

Fonte: Próprio Autor

O desenvolvimento do aplicativo foi conduzido utilizando o *framework* React Native, detalhado com mais profundidade na Seção 3.4. A API responsável pela comunicação entre a aplicação e a loja de modelos foi implementada em Python, utilizando a biblioteca

Flask para garantir a eficácia da comunicação. Os modelos utilizados neste trabalho foram previamente construídos e validados em trabalhos paralelos a este, sendo meramente objeto de experimento para o presente trabalho.

Com o propósito de simplificar a aplicação de modelos de aprendizado de máquina diretamente em dispositivos móveis, a biblioteca Playtorch foi incorporada ao aplicativo. Essa biblioteca permite a integração do PyTorch em aplicativos desenvolvidos em React Native, o *framework* utilizado no desenvolvimento do aplicativo em questão. Além de facilitar essa integração, a Playtorch executa um processo de otimização do modelo, visando reduzir o consumo de recursos. Esse aspecto é particularmente relevante, considerando as limitações de recursos em dispositivos móveis em comparação com computadores desktop.

5.2 Arquitetura da Aplicação Móvel

Esta seção tem por objetivo elucidar e detalhar o funcionamento do aplicativo desenvolvido. Inicialmente, são apresentados os requisitos funcionais essenciais para o adequado desempenho do sistema:

- O sistema possui cadastro de novos usuários.
- Os usuários têm a capacidade de modificar suas senhas diretamente no aplicativo.
- A aplicação requer uma conexão inicial com a Internet.
- Usuário da IAaaS precisa estar previamente autenticado.
- É necessário obter permissão de acesso à câmera e à galeria de fotos do dispositivo móvel.
- A aplicação realiza a tarefa de processamento e classificação de imagens.
- São apresentados os modelos de Inteligência Artificial disponíveis na loja de modelos e no banco de dados local do dispositivo.

A Figura 6 detalha a maneira como ocorre a sequencia logica de funcionamento do aplicativo. Realizando um comparativo entre a imagem e os requisitos temos que o requisito de autenticação do usuário é representado no diagrama apresentado na Figura 6. O usuário só poderá avançar para a utilização do software se estiver devidamente cadastrado e suas informações de login forem verificadas com sucesso. Caso o usuário não possua um login, o sistema aborda a necessidade de permitir o cadastro de novos usuários, conforme ilustrado na parte esquerda da primeira decisão.

Em seguida, o sistema aborda o requisito de listar todos os modelos de aprendizado de máquina disponíveis para uso pelo usuário. Esse aspecto é visualizado de maneira

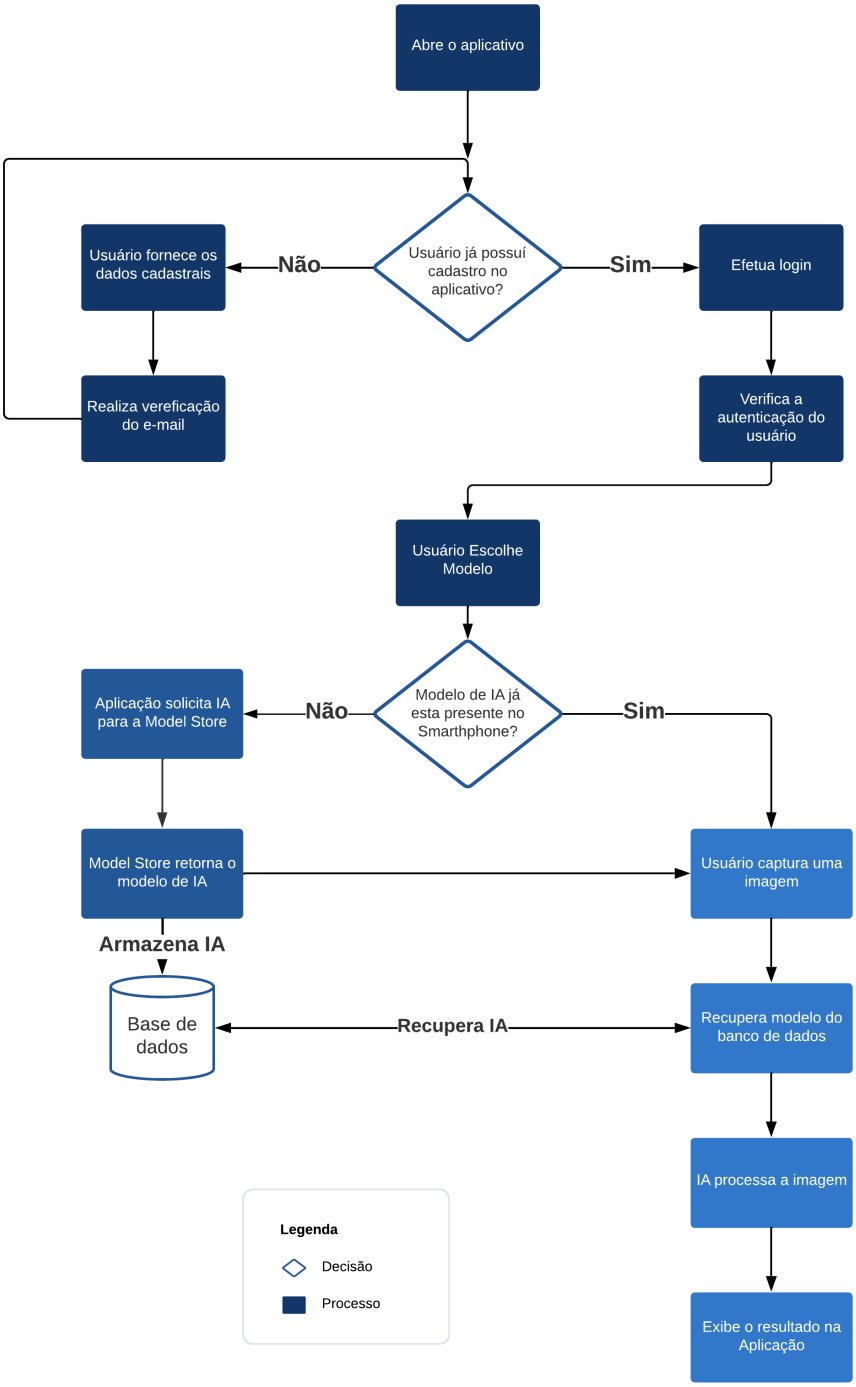

Figura 6 – Fluxograma representando a sequencia lógica da aplicação

.

Fonte: Próprio Autor

mais concreta na Figura 9, especialmente na terceira tela, onde é possível realizar pesquisas pelo nome na loja de modelos. Além disso, caso o usuário tenha utilizado modelos anteriormente, ele pode procurar na biblioteca o modelo desejado.

O fluxograma apresentado na Figura 6 delineia a sequência lógica da classificação de imagem. Nesse processo, o usuário seleciona um modelo de classificação de imagem, e a aplicação verifica a presença desse modelo no banco de dados local. Caso não esteja disponível localmente, o sistema envia uma requisição por meio de uma API para a *AI Models Store*, solicitando o modelo desejado. A resposta da nuvem é então recebida, contendo o algoritmo de aprendizado de máquina correspondente, que é armazenado no banco de dados local do dispositivo.

Após essa etapa, o usuário inicia o procedimento de envio da imagem, e o aplicativo dá início ao processo de classificação do problema. Uma vez obtido o resultado, este é exibido na tela para o usuário final.

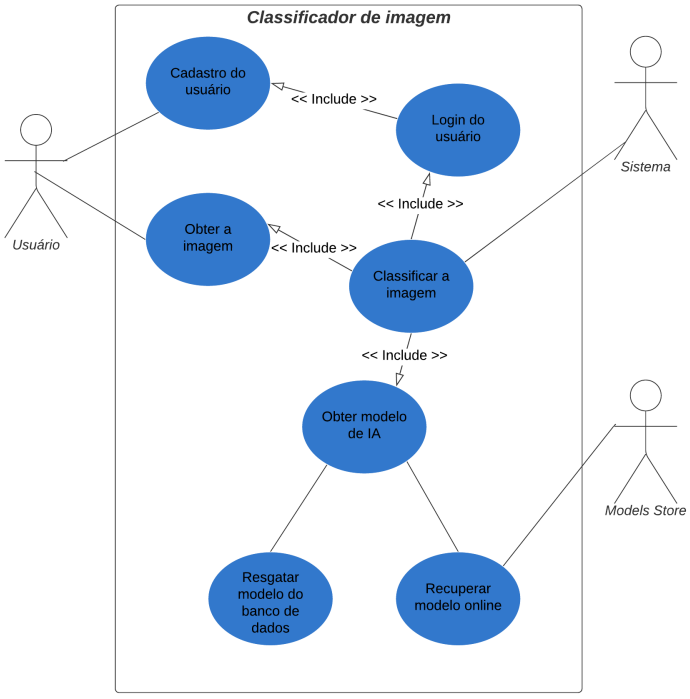

Figura 7 – Imagem mostrando o diagrama do caso de uso

.

Fonte: Próprio Autor

A Figura 7 apresenta o diagrama de caso de uso do presente projeto, no qual torna possível a visualização dos atores (Usuário, Sistema e *Models Store*) apresentando a forma com que cada um deles interage na aplicação. É perceptível que o usuário atua realizando o seu próprio cadastro e enviando a imagem para a classificação. A *Models Store* é acionada somente quando é necessário realizar a recuperação de um modelo na nuvem.

Um importante aspecto do diagrama representado na ligação entre dois processos que possui a palavra *"Include"* significa que necessariamente este processo depende do outro, em outras palavras o primeiro não consegue ser executado sem que ocorra a utilização de maneira correta do outro processo.

Para complementar o diagrama de caso de uso ilustrado na Figura 7, propõe-se o diagrama de sequência representadona Figura 8 que detalha o passo a passo do processo de classificar uma imagem dentro do aplicativo. As linhas tracejadas na vertical representam os atores sendo que nesta linha ficam em destaque no momento em que são utilizados no processo. Neste diagrama é possível perceber mais detalhadamente cada interação entre os atores e tornando viável visualizar o tempo de vida e atuação de cada um deles no *software*.

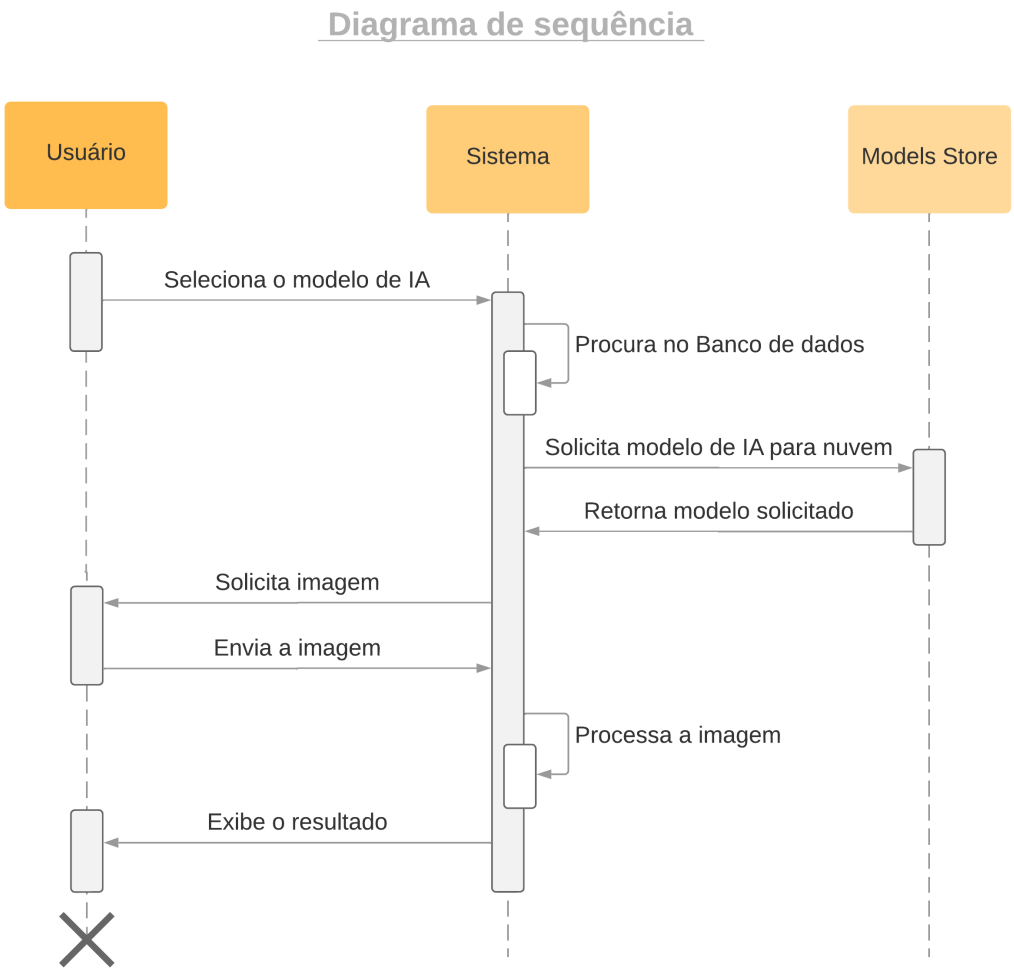

Figura 8 – Imagem mostrando o diagrama de sequência

.

Fonte: Próprio Autor

5.3 Prototipação

Esta seção, apresentamos uma análise detalhada do processo de prototipação, destacando as telas cuidadosamente desenvolvidas para o aplicativo, juntamente com suas principais funcionalidades. Nosso objetivo é fornecer uma demonstração abrangente do funcionamento do aplicativo, oferecendo informações precisas sobre sua interface e operacionalidade.

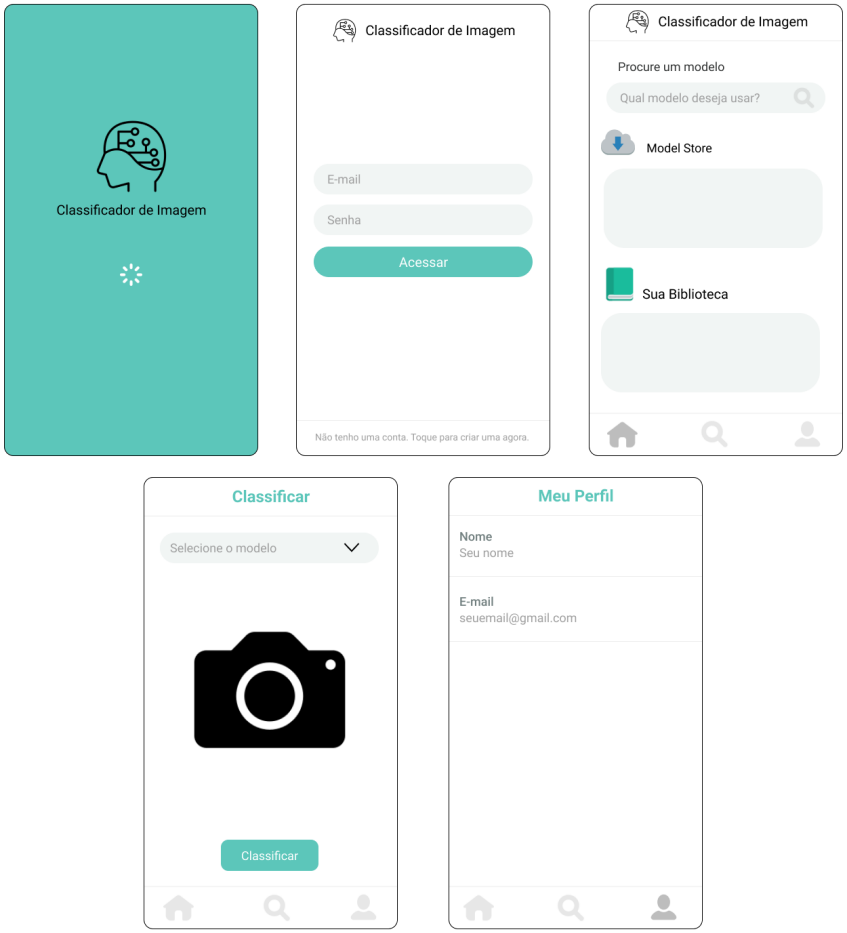

Figura 9 – Proposta de Telas para a Solução Proposta.

.

Fonte: Próprio Autor

A Figura 9 apresenta as telas do aplicativo, que incluem a Tela Inicial, Autenticação, Home, Classificar Imagem e Perfil, respectivamente. A Tela Inicial é exibida quando o usuário abre o aplicativo, durante o carregamento de suas dependências. Em seguida, surge a Tela de Autenticação, que requer que o usuário faça login ou crie uma conta para acessar o aplicativo. A Tela Home representa a interface principal, permitindo o *download* de modelos de aprendizado de máquina da nuvem e a visualização dos já presentes no aplicativo.

A tela subsequente é dedicada à escolha do modelo de IA e à captura de uma

fotografia, na qual o algoritmo realizará a classificação e fornecerá o diagnóstico correspondente. Por fim, na última tela, denominada Perfil, são apresentadas as informações cadastrais do usuário.

5.4 Gerenciamento dos Modelos de Aprendizado de Máquina

Os modelos empregados no projeto foram treinados por meio da biblioteca PyTorch, utilizando a base de dados USK Coffee Febriana et al. (2022) composta por 8000 imagens categorizadas em quatro tipos distintos: “premium”, “peaberry”, “longberry” e “defect”, representadas respectivamente em Figura 10, Figura 11, Figura 12 e Figura 13, cada um representando diferentes características de frutos de café. Cada modelo de classificação foi treinado utilizando uma arquitetura de rede neural convolucional predefinida pelo PyTorch, incluindo ResNet50 He et al. (2015), MobileNetv3 Howard et al. (2019), EfficientNet Tan, Tan e Le (2019) e AlexNet Krizhevsky et al. (2012).

Figura 10 – Ilustração representativa da classe Premium.

.

Fonte: (FEBRIANA et al., 2022)

Para viabilizar a implementação desses modelos em dispositivos móveis, foi necessária uma etapa de conversão dos modelos treinados no PyTorch para torná-los compatíveis com a biblioteca Playtorch. Essa conversão foi realizada por meio de um script em Python, que inicialmente carrega os modelos treinados no PyTorch. Em seguida, a biblioteca Torch Utils é empregada, utilizando a função de otimização para dispositivos móveis. Dessa maneira, o modelo resultante torna-se compatível com o Playtorch, possibilitando sua eficiente utilização em dispositivos móveis.

Figura 11 – Ilustração representativa da classe Peaberry.

.

Fonte: (FEBRIANA et al., 2022)

Figura 12 – Ilustração representativa da classe Longberry.

.

Fonte: (FEBRIANA et al., 2022)

Figura 13 – Ilustração representativa da classe Defect.

.

Fonte: (FEBRIANA et al., 2022)

5.5 Cenário Experimental

Nesta seção, discutiremos a metodologia adotada para a execução dos testes. Utilizamos um dispositivo Motorola Edge 30 Pro, equipado com um sistema operacional Android, um processador octa-core de 2.2GHz e 12GB de memória RAM, para a instância e execução do aplicativo. Além disso, uma máquina com uma placa de vídeo RTX4060TI foi empregada para a implementação da API de classificação.

A comunicação entre o aplicativo e a máquina ocorreu por meio de requisições web. Para facilitar esse processo, desenvolvemos uma API no computador, utilizando o Flask para garantir sua funcionalidade. Nesse contexto, os modelos foram armazenados no desktop, e quando o aplicativo enviava uma requisição por meio da rede local, a API respondia disponibilizando o modelo desejado.

A mensuração do tempo de *download* foi realizada no ambiente React Native, utilizando a função *Date*. Essa função foi instanciada no início da requisição e novamente ao término da mesma. O tempo de *download* foi calculado como a diferença entre a data final e a inicial, expresso em milissegundos.

Para assegurar a confiabilidade dos testes, foram executadas dez requisições de *download* para cada modelo utilizado. Ao final, a média aritmética foi calculada, proporcionando um tempo de *download* representativo da realidade.

6 Resultados

Este capítulo apresenta os resultados quantitativos e qualitativos do trabalho desenvolvido. Inicialmente, discute-se o tempo de predição de um modelo de aprendizado de máquina, comparando sua execução no dispositivo móvel e em uma API hospedada em um servidor de alto desempenho.

Em seguida, aborda-se de forma quantitativa o consumo de memória, destacando o impacto de cada um dos quatro modelos em relação às predições realizadas. Além disso, é fornecido uma comparação do consumo de memória associado à localização da predição. Por fim, da-se inicio a uma reflexão sobre o tempo de *download* de cada modelo, identificando o momento em que ocorre a solicitação do usuário na loja de modelos, desencadeando o processo de *download*.

6.1 Tempo de predição

Visando aprimorar a experiência do usuário, um elemento crucial a ser considerado é o tempo de espera necessário para executar uma ação específica. No contexto deste trabalho, em que a ação principal consiste na predição de uma imagem, torna-se essencial que esse processo ocorra de maneira ágil. Dessa forma, foram conduzidos testes utilizando duas abordagens distintas: a primeira envolvendo a predição diretamente no dispositivo móvel e a segunda fazendo uso de uma API em um servidor de alta performance.

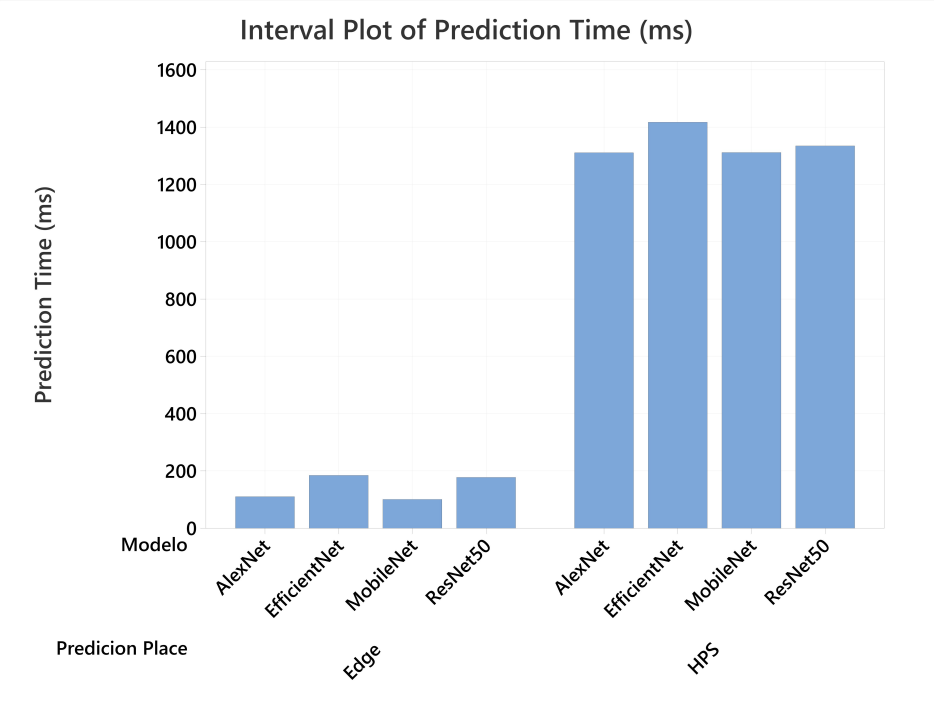

Figura 14 – Gráfico do tempo de predição por modelo

.

Fonte: Próprio Autor

Após a realização dos testes, a Figura 14 representa a análise do tempo demandado para a tarefa de predição em diferentes modelos, categorizados por local de execução onde, no eixo y, o tempo de predição é expresso em segundos. Observa-se que o servidor exibiu um tempo de predição significativamente superior em comparação com os testes conduzidos no dispositivo móvel do usuário. Essa disparidade é atribuída ao tempo de transmissão da imagem pela rede no caso do servidor, evidenciando a importância de considerar esse fator na otimização do desempenho do sistema.

6.2 Consumo de Memória

A eficiência da aplicação está intrinsecamente ligada à quantidade de memória consumida, sendo ideal exigir o mínimo possível do dispositivo. A análise do consumo de memória, evidenciada no Figura 15, revela que o modelo AlexNet demonstrou o menor consumo durante os testes, enquanto, em contrapartida, o modelo ResNet50 demandou a maior alocação de memória.

Diante dessas constatações, torna-se inclinada a preferência por modelos que utilizam a arquitetura de rede neural convolucional, fornecida pelo AlexNet, para aplicações móveis, considerando sua eficiência ao manter um consumo de memória menor em relação aos outros modelos.

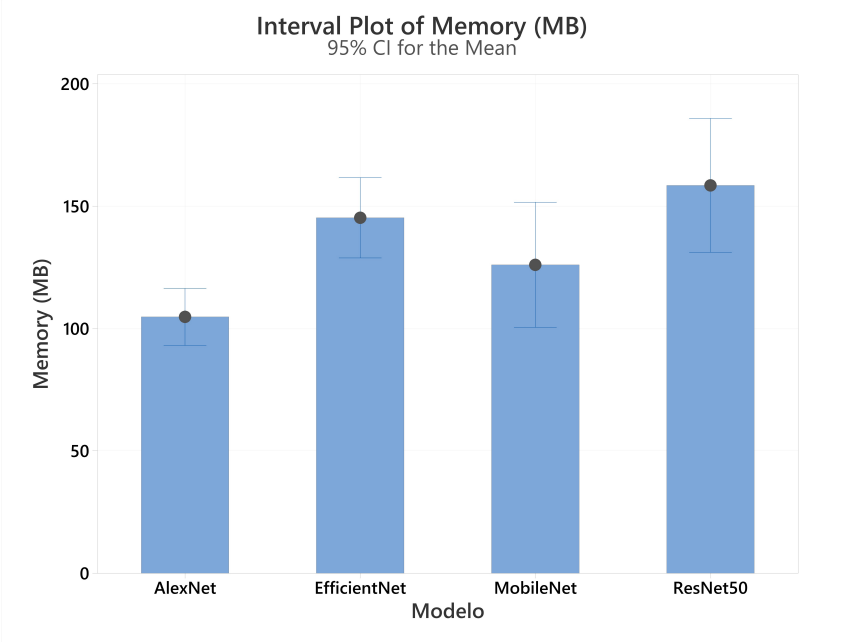

Figura 15 – Gráfico do consumo de memória por modelo no dispositivo móvel

.

Fonte: Próprio Autor

Um aspecto interessante de abordar no contexto do consumo de memória é a comparação entre dois locais de predição: no próprio dispositivo e em um Servidor de Alta Performance (HPS). A Figura 16 ilustra os resultados dos testes de desempenho, destacando o consumo de memória em cada cenário.

Ao analisar a Figura 16, torna-se evidente que o consumo de memória foi significativamente menor no dispositivo móvel em comparação com o HPS. Essa redução pode ser atribuída ao processo de otimização para dispositivos móveis, conforme descrito na Seção 5.4. Esse processo é executado em cada modelo antes de ser utilizado no dispositivo, contribuindo para uma eficiência notável.

A importância desse processo de otimização torna-se evidente ao considerarmos que o consumo médio de memória através da API foi de aproximadamente 160MB, enquanto no dispositivo móvel foi de apenas 100MB. Isso representa uma economia de

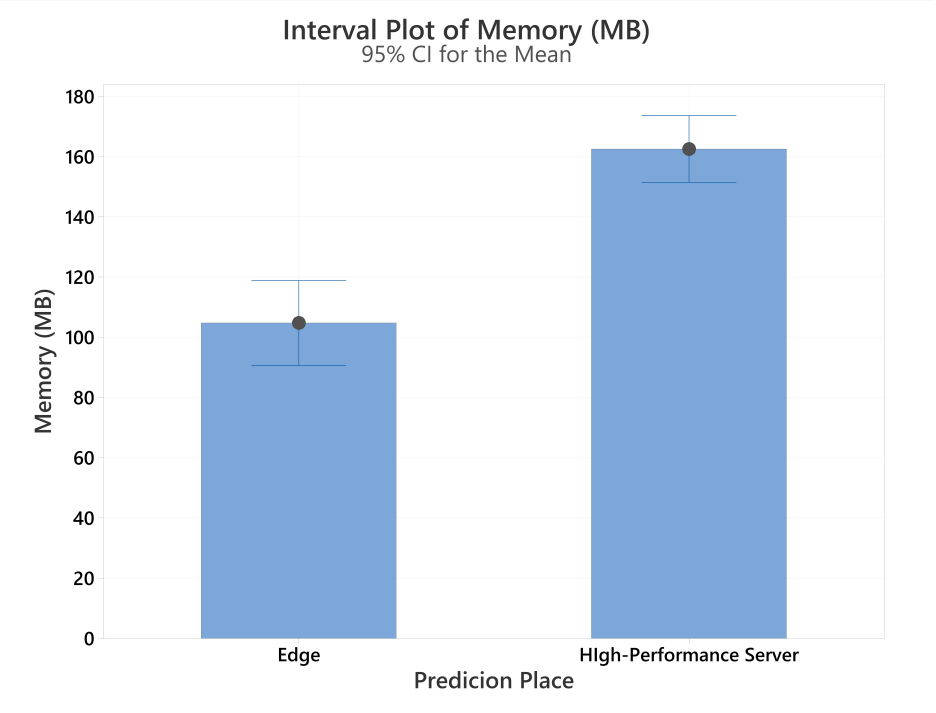

Figura 16 – Gráfico do consumo de memória por local de predição

.

Fonte: Próprio Autor

memória de quase 40%, destacando a relevância prática e impactante da otimização no desempenho geral do sistema.

6.3 Tempo de *Download*

Esta seção abordará o tempo de *download*, que ocorre sempre que o usuário solicita a aquisição de um novo modelo de aprendizado de máquina para a loja de modelos. A relevância desse aspecto está intrinsecamente ligada à experiência do usuário, uma vez que é indesejável que o tempo de espera para utilizar o aplicativo seja prolongado.

Na Figura 17, apresenta-se o tempo de *download*, em milissegundos, para cada modelo utilizado nos testes. Observa-se que o modelo AlexNet demonstra um tempo de *download* significativamente superior em comparação aos demais modelos.

Conforme discutido na Seção 6.2, reconhecemos a extrema importância do recurso memória. Entretanto, ao analisarmos conjuntamente o tempo de *download* e o consumo

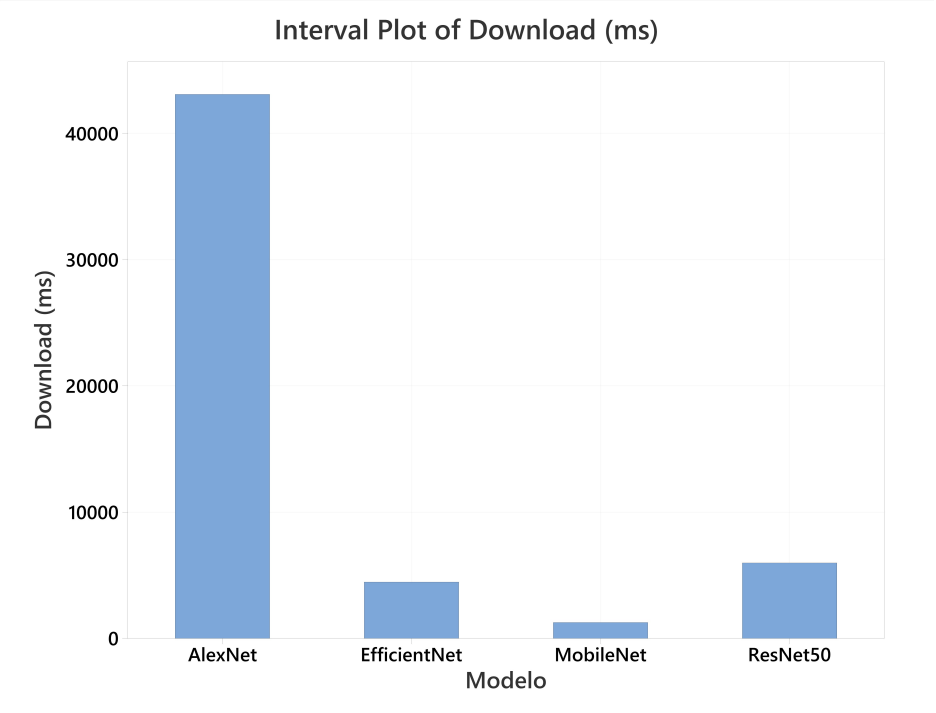

Figura 17 – Gráfico do tempo de *download* por modelo

.

Fonte: Próprio Autor

de memória, evidencia-se que o AlexNet apresenta um baixo consumo de memória, mas seu tempo de *download* é substancial. Nesse contexto comparativo, surge a opção pelo modelo MobileNet, visto que seu consumo de memória é apenas ligeiramente superior ao da AlexNet. No entanto, seu tempo de *download* é consideravelmente menor, tornando-o uma escolha mais vantajosa.

7 Conclusão

A crescente popularidade do uso de dispositivos móveis, impulsionada pela praticidade e acessibilidade que oferecem, resultou em um aumento significativo no número de pessoas que possuem seus próprios celulares. Esse fenômeno, por sua vez, gerou uma expansão no volume de imagens capturadas e armazenadas, proporcionando uma oportunidade substancial para a área de visão computacional analisá-las.

Diante da diversidade inerente à área de visão computacional, surge a possibilidade de preencher uma lacuna por meio do desenvolvimento de um aplicativo que adote o paradigma de IAaaS. Tal abordagem permite a classificação de diversos tipos de imagens em um único aplicativo, proporcionando uma solução versátil e eficiente.

Este estudo concentrou-se no desenvolvimento desse aplicativo, seguido pela realização de testes de desempenho, conforme discutido no Capítulo 6. Os resultados desses testes revelam que a utilização de modelos de aprendizado de máquina para a predição de imagens, executados localmente no dispositivo móvel, demonstrou ser mais vantajosa. Essa abordagem resulta em tempos de predição mais rápidos e menor consumo de recursos.

Apesar da desvantagem inerente ao método de classificação local, que envolve o *download* do modelo, é crucial ressaltar que esse processo ocorre apenas uma vez para cada modelo desejado pelo usuário. Considerando as vantagens previamente mencionadas, a opção pela classificação local no dispositivo ainda se mostra atrativa, superando o obstáculo do *download* inicial do modelo.

Os desdobramentos potenciais para trabalhos futuros compreendem aprimoramentos no aplicativo, notadamente a incorporação de alternativas de autenticação adicionais, como integração com plataformas de autenticação como Gmail e Facebook. Ademais, a expansão da variedade de modelos de classificação de imagem disponíveis na loja de modelos do aplicativo figura como uma consideração relevante. A investigação e implementação de métodos mais robustos para o armazenamento seguro de dados constituem outra vertente significativa a ser explorada. Além disso, a avaliação sistemática da interface do aplicativo, visando a otimização da experiência do usuário, emerge como um componente essencial para futuros desenvolvimentos.

Referências

ABELL, T. **Why cloud computing is a key enabler for Digital Government**. 2021. Disponível em: <https://development.asia/explainer/why-cloud-computing-key-enabler-digital-government>.

BAEK, J.; KIM, S.; SHIN, D. Anomaly detection for an oral health care application using one class yolov3. **J. Korean Soc. Ind. Appl. Math. Vol**, v. 26, n. 4, p. 310–322, 2022.

BELL, J. Machine learning hands-on for developers and technical professional, vol. 134, no. 4. **Indianapolis, Indiana**, 2007.

BHATTACHARJEE, N. Automated dental cavity detection system using deep learning and explainable ai. In: AMERICAN MEDICAL INFORMATICS ASSOCIATION. **AMIA Annual Symposium Proceedings**. [S.l.], 2022. v. 2022, p. 140.

CARDON, A.; MÜLLER, D. N.; NAVAUX, P. Introdução às redes neurais artificiais. **Porto Alegre**, 1994.

DANIELSSON, W. React native application development. **Linköpings universitet, Swedia**, v. 10, n. 4, p. 10, 2016.

DING, B. et al. Detection of dental caries in oral photographs taken by mobile phones based on the yolov3 algorithm. **Annals of Translational Medicine**, AME Publications, v. 9, n. 21, 2021.

EISENMAN, B. **Learning react native: Building native mobile apps with JavaScript**. O'REILLY: "O'Reilly Media, Inc.", 2015.

ESGARIO, J. G. et al. An app to assist farmers in the identification of diseases and pests of coffee leaves using deep learning. **Information Processing in Agriculture**, Elsevier, v. 9, n. 1, p. 38–47, 2022.

FEBRIANA, A. et al. Usk-coffee dataset: A multi-class green arabica coffee bean dataset for deep learning. **International Conference on Cybernetics and Computational Intelligence**, IEEE, 2022.

FLECK, L. et al. Redes neurais artificiais: Princípios básicos. **Revista Eletrônica Científica Inovação e Tecnologia**, v. 1, n. 13, p. 47–57, 2016.

FRADKOV, A. L. Early history of machine learning. **IFAC-PapersOnLine**, Elsevier, v. 53, n. 2, p. 1385–1390, 2020.

HAYKIN, S. **Redes neurais: princípios e prática**. ARTMED Editora: Bookman Editora, 2001.

HE, K. et al. **ResNet-50**. 2015. Disponível em: <https://arxiv.org/abs/1512.03385>.

HOWARD, A. et al. **MobileNetV3**. 2019. Disponível em: <https://arxiv.org/abs/1905.02244>.

JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. **Electronic Markets**, Springer, v. 31, n. 3, p. 685–695, 2021.

KIM, D. et al. A smart home dental care system: integration of deep learning, image sensors, and mobile controller. **Journal of Ambient Intelligence and Humanized Computing**, Springer, p. 1–9, 2021.

KRIZHEVSKY, A. et al. **AlexNet**. 2012. Disponível em: <https://pytorch.org/hub/pytorch_vision_alexnet/>.

MARKIEWICZ, T.; ZHENG, J. **Getting Started with Artificial Intelligence**. [S.l.]: O'Reilly Media, Incorporated, 2020.

MOREIRA, R. et al. Agrolens: A low-cost and green-friendly smart farm architecture to support real-time leaf disease diagnostics. **Internet of Things**, Elsevier, v. 19, p. 100570, 2022.

NUNES, V. G. C. Avaliação de abordagens e definição de diretrizes para o desenvolvimento de aplicativos mobile: um relato de experiência utilizando aabordagem não nativa de desenvolvimento. Universidade Federal Rural do Semi-Árido, 2021.

OFOEDA, J.; BOATENG, R.; EFFAH, J. Application programming interface (api) research: A review of the past to inform the future. **International Journal of Enterprise Information Systems (IJEIS)**, IGI Global, v. 15, n. 3, p. 76–95, 2019.

PEDROSA, P. H.; NOGUEIRA, T. Computação em nuvem. **Acesso em**, v. 6, 2011.

PEREIRA, T. M.; SACILOTTI, A. C.; JÚNIOR, J. R. M. Computação em nuvem: plataforma como serviço. **MARTINS, Ernane Rosa. Fundamentos da Ciência da Computação**, v. 2, p. 116–125, 2019.

RAUBER, T. W. Redes neurais artificiais. **Universidade Federal do Espírito Santo**, v. 29, 2005.

RAY, S. A quick review of machine learning algorithms. p. 35–39, 2019.

RODRIGUES MOREIRA, L. F. et al. An artificial intelligence-as-a-service architecture for deep learning model embodiment on low-cost devices: A case study of covid-19 diagnosis. **Applied Soft Computing**, Elsevier, v. 134, p. 110014, 2023.

SCABINI, L. F. et al. Angular descriptors of complex networks: A novel approach for boundary shape analysis. **Expert Systems with Applications**, Elsevier, v. 89, p. 362–373, 2017.

SUNYAEV, A. Cloud computing. In: _____. **Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies**. Cham: Springer International Publishing, 2020. p. 195–236. ISBN 978-3-030-34957-8. Disponível em: <https://doi.org/10.1007/978-3-030-34957-8_7>.

TAN, M.; TAN, Q.; LE. **EfficientNet**. 2019. Disponível em: <https://arxiv.org/abs/1905.11946>.

THANH, M. T. G. et al. Deep learning application in dental caries detection using intraoral photos taken by smartphones. **Applied Sciences**, v. 12, n. 11, 2022. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/12/11/5504>.

VALENTE, J. **Mais de 5 bilhões de pessoas usam aparelho celular, revela pesquisa**. 2019. Disponível em: <https://agenciabrasil.ebc.com.br/geral/noticia/2019-09/mais-de-5-bilhoes-de-pessoas-usam-aparelho-celular-revela-pesquisa>.

ZENG, W. et al. Lightweight dense-scale network (ldsnet) for corn leaf disease identification. **Computers and Electronics in Agriculture**, Elsevier, v. 197, p. 106943, 2022.

Apêndices

APÊNDICE A – Artefatos de *Software*: Mecanismo de Predição

O presente artefato de código refere-se a operacionalização da predição de imagens, destacando a execução do processo no próprio dispositivo

```
import {torch, torchvision, media } from 'react-native-pytorch-core';
import * as ImageNetClasses from './ImageNetClasses.json';

const T = torchvision.transforms;
const MODEL_URL = 'https://github.com/lucasnardelli/testeDownload/raw/
    main/final.ptl';
let model = null;

export default async function classifyImage(image, download) {

  const width = image.getWidth();
  const height = image.getHeight();
  const blob = media.toBlob(image);

  let tensor = torch.fromBlob(blob, [height, width, 3]);
  tensor = tensor.permute([2, 0, 1]);
  tensor = tensor.div(255);

  const resize = T.resize(224);
  tensor = resize(tensor);

  const normalize = T.normalize((0.485, 0.456, 0.406), (0.229, 0.224,
    0.225));
  tensor = normalize(tensor);

  tensor = tensor.unsqueeze(0);

  model = download

  const output = await model.forward(tensor);
  const maxIdx = output.argmax().item();

  return ImageNetClasses[maxIdx]
}
```

Listing A.1 – Código de predição no aplicativo

APÊNDICE B – Artefatos de *Software*: Mecanismo de Conversão de Modelos

O código em questão concentra-se na conversão de um modelo desenvolvido no PyTorch para a biblioteca Playtorch, delineando detalhadamente o procedimento associado a essa transição.

```
1  import torch
2  import torchvision
3  from torch.utils.mobile_optimizer import optimize_for_mobile
4  import torchvision.models as models
5
6  model = models.squeezenet1_0(num_classes=4)
7
8  model.load_state_dict(torch.load('/home/nrd/Desktop/squeezenet.pth',
       map_location=torch.device('cpu')))
9
10 model.eval()
11
12 scripted_model = torch.jit.script(model)
13 optimized_model = optimize_for_mobile(scripted_model)
14 optimized_model._save_for_lite_interpreter("/home/nrd/Desktop/final.ptl"
       )
15
16 print("model successfully exported")
```

Listing B.1 – Código de conversão dos modelos